Overview

This taks involves extracting entity values from product images as part of a hackathon. The task was to predict values such as weight, volume, voltage, etc., based on image data. I utilized EasyOCR for text extraction and regular expressions (regex) for pattern matching to predict the entity values.

Project Structure

images/: Contains the downloaded images used for prediction.

src/: Contains utility scripts such as constants.py, sanity.py, and the main script responsible for prediction.

dataset/: Contains the train and test CSV files with product metadata.

test_out.csv: Final output file containing predicted values.

Key Libraries

EasyOCR: A Python OCR tool used for extracting text from images.

Regex (Regular Expressions): For identifying and extracting specific patterns from the text data.

Source Code Explanation

1. Importing Libraries

```
  import cv2
  import pandas as pd
  import numpy as np
  from PIL import Image
  import matplotlib.pyplot as plt
  import easyocr
  from src import constants
  import re
  import csv
  import os
  import subprocess
```

2. Image preprocessing

```
def preprocess_image(image_path):
img = cv2.imread(image_path)
resized_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
return resized_img
```

3. Extraction of key values

```
  reader = easyocr.Reader(['en'])
  def extract_text(image_path):
    preprocessed_img = preprocess_image(image_path)
    result = reader.readtext(preprocessed_img)

    extracted_text = " ".join([text[1] for text in result])
    return extracted_text
```

4. Unit abbreviation mapping

```
  unit_abbreviation_mapping = {
      "cm": "centimetre",
      "centimeter": "centimetre",
      "mm": "millimetre",
      "millimeter": "millimetre",
      "kg": "kilogram",
      "kilogramme": "kilogram",
      "g": "gram",
      "gm": "gram",
      "gramme": "gram",
      "ml": "millilitre",
      "milliliter": "millilitre",
      "l": "litre",
      "lt": "litre",
      "m": "metre",
      "meter": "metre",
      "kv": "kilovolt",
      "v":"volt",
      "kw": "kilowatt",
      "w":"watt",
      "oz": "ounce",
      "ft": "foot",
      "foot": "foot",
      "in": "inch",
      "inch": "inch",
      "lbs":"pound",

  }
```

5. OCR correction

```
ocr_correction ={"S":"5", "O":"0"}
def correct_ocr_errors(text):
    for incorrect,correct in ocr_correction.items():
        text = text.replace(incorrect,correct)
    return text

def map_abbreviation_to_unit(unit):
    unit = unit.lower().strip()
    return unit_abbreviation_mapping.get(unit,unit)
```

6.Extraction and mapping

```
def map_extracted_text_to_values(extracted_text):
    extracted_text = correct_ocr_errors(extracted_text)
    extracted_text = re.sub(r'\s+', ' ', extracted_text).strip()
    pattern= r"(\d+(?:\.\d+)?)\s*(\w+)"
    matches = re.findall(pattern,extracted_text)

    mapped_values = []
    for value,unit in matches:
        full_unit = map_abbreviation_to_unit(unit)
        if full_unit in constants.allowed_units:
            return f"{float(value)} {full_unit}"
```

7. Prediction Flow and output saving into test_out.csv

```
def generate_predictions(image_folder_path,output_csv):
      output_folder = 'output'
      if not os.path.exists(output_folder):
          os.makedirs(output_folder)
      output_path = os.path.join(output_folder,output_csv)

      with open(output_path, mode = 'w', newline = '') as file:
          writer = csv.writer(file)
          writer.writerow(['index','prediction'])


          for index,filename in enumerate(os.listdir(image_folder_path)):
              print(f"processing image: {index+1} , file : {filename} ")
              if not os.path.exists(image_folder_path):
                  print(f"image not found for index: {index}")
                  prediction = ""
              else:
                  image_path = os.path.join(image_folder_path,filename)
                  extracted_text = extract_text(image_path)
                  prediction = map_extracted_text_to_values(extracted_text)


              writer.writerow([index,prediction])
      print(f"output csv saved as {output_csv}")
```

Sanity Check

The sanity checker script (sanity.py) provided by the hackathon organizers was used to validate the output format and ensure correctness before submission.

Conclusion

This approach effectively combines OCR and regex to automate the extraction of specific entity values from product images. Future improvements could include training a CNN model to better identify and classify image features for more complex entity types.